

ラズパイでFPGA

2018/5/17

パナソニックソリューションテクノロジー株式会社

川村 信

自己紹介

名前 川村 信（かわむら まこと）
所属 パナソニックソリューションテクノロジー（株）

仕事

昨年9月よりデータ分析の仕事をしています。
現在、SQLと格闘中

以前の仕事

テレビ用マイコンの開発	⇒	デジタル回路設計
デジタルテレビ用SoC	⇒	マイクロコード
	⇒	自動実機検査装置
	⇒	DDR3メモリI/F
ToFセンサ	⇒	USB3.0
	⇒	FPGA（画像処理）

その他 SLAM、ROS、PYTHON、動線分析などなど

休日 5歳の息子とお出かけ。

ラズパイで何かを….

誰も使っていないハードに挑戦するぞ！！



ラズパイでFPGA！！



調べてみると…

ラズパイ用のFPGAってあんまりない…

FPGAとは

Wikipediaより

FPGA（[英](#): field-programmable gate array）は、製造後に購入者や設計者が構成を設定できる[集積回路](#)であり、広義にはPLD（[プログラマブルロジックデバイス](#)）の一種である。現場でプログラム可能な[ゲートアレイ](#)であることから、このように呼ばれている。



個人がデジタル回路を作ることができる半導体のこと。

- CPUが作れる。
- インターフェイスが作れる。
- 映像回路も作れる。
- 何でも？作れる。



長所
短所

- CPUが命令を順番に処理するのに対し、FPGAなら、並列に計算する回路を作れば高速！
- CPUよりクロックが遅い。
- 設計が難しい。
- タイミング設計が必要。

FleaFPGA Ohm Board FleaFPGA Ohm Board

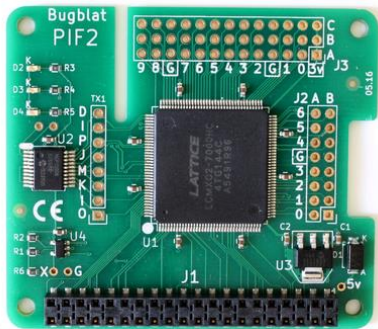


ラティスセミコンダクターのFPGA「ECP5」を搭載したFPGA開発ボード「FleaFPGA Ohm Board」が、Indiegogoにて出資を募っている。

https://fabcross.jp/news/2017/20171005_latticeecp5_fpgadevboard_fleafpga.html

まだ売っていない

Bugblat社Raspberry Pi FPGAアドオンボード



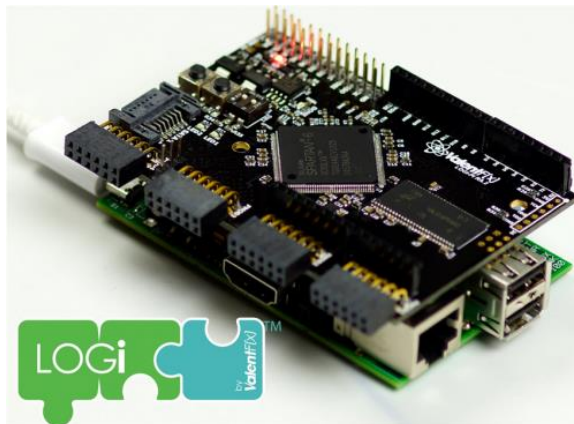
XO2: a Lattice Semiconductor MachXO2 FPGA (details at Lattice)

PIF_2: a powerful FPGA (7000 LUTs) for the Pi 1B+, Pi 2B, and Pi 3B.

<http://www.bugblat.com/products/pif/>

\$39.99

FPGA Development Board for the RASPBERRY PI



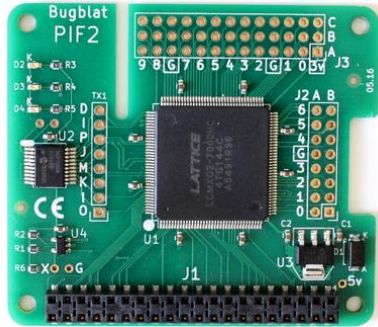
FPGA: Spartan 6 LX9 – TQFP144 Package - XC6SLX9-2TQG144C

Plug and play interfacing the Raspberry Pi
Arduino Shield expansion allowing for more than 200 existing plug in hardware modules

<https://www.element14.com/community/docs/DOC-69210/1/fpga-development-board-for-the-raspberry-pi>

Price: £69.22 **Out of Stock**

Bugblat社Raspberry Pi FPGAアドオンボード



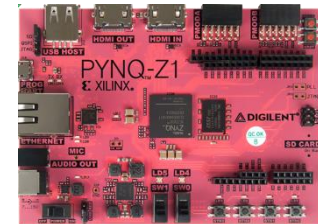
XO2: a Lattice Semiconductor MachXO2 FPGA (details at Lattice)
PIF_2: a powerful FPGA (7000 LUTs) for the Pi 1B+, Pi 2B, and Pi 3B.

<http://www.bugblat.com/products/pif/>

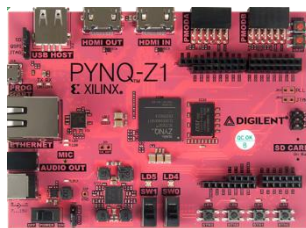
これに決定！

GW前に注文 → → → → 届かない。。。 → GW突入！

→ GW開けた。。。 → 来ないので別のものに手を出す。
(Xilinx PYNQ-Z1 FPGAボード)



→ 3日後 ボード到着(早い！！) → 翌日 ボード到着



やっと届いたけど…… 動かない。



~~で、現在デバッグ & 不具合原因究明中（涙）~~



動きました！
（後述）

以下デバッグ状況

起動するためには、まず、ラズパイがボードを認識する必要がある。



プログラム（piffing.py : python）が付属



I2cとSPIを有効化



実行すると…ライブラリがない等エラー…



ライブラリなどをインストールして再挑戦



```
=====hello=====
Configuration file is /home/pi/test_impl1.jed
Using pif library version: 'libpif,Sep 15 2016,13:35:35'

XO2 Device ID: 00000000 recognised ID!
XO2 Trace ID : 00.00.00.00_00.00.00.00
XO2 usercode from Flash: 00.00.00.00
XO2 usercode from SRAM : 00.00.00.00
JEDEC file is /home/pi/test_impl1.jed
starting to read JEDEC file
JEDEC file does not match FPGA

FPGA is XO2-256HC

JEDEC identifies as "NOTE DEVICE NAME:    LCMXO2-7000HC-4TQFP144*"

=====bye=====
```

デバイスIDが読めていない！

piffind.py


```
##-----
def showDeviceID(handle):

    dw = c_ulong(0xdeadbeef)
    res = pifglobs.pif.pifGetDeviceIdCode(handle, byref(dw))
    if (res == 0):
        print("¥nread ID code failed¥n")
        return "failed"

    deviceID = dw.value
    print('X02 Device ID: %08x' % deviceID),
```

このデバイスIDをとっている関数が
どのような処理をしているか追いかける！！


pifwrap.cpp



```
int pifGetDeviceIdCode(pifHandle h, uint32_t* v) {
    return pPif->getDeviceIdCode(*v);
}

int pifGetStatusReg(pifHandle h, uint32_t* v) {
    return pPif->getStatusReg(*v);
}
```

pif.cpp

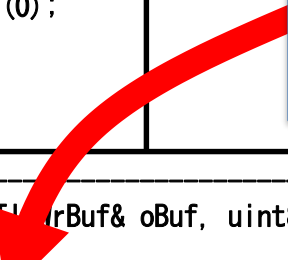


```
//-----
bool Tpif::getDeviceIdCode(uint32_t& v) {
    v = 0;
    TllWrBuf oBuf;
    oBuf.byte(READ_DEVICE_ID_CODE).byte(0).byte(0).byte(0);

    uint8_t p[4];
    bool ok = _cfgWriteRead(oBuf, p, 4);
    v = _dwordBE(p);
    return ok;
}
```

SPI I/Fで
デバイスIDを取得している
ことが判明。

⇒SPI I/Fをチェックする



```
//-----
bool Tpif::_cfgWriteRead(TllWrBuf& oBuf, uint8_t *pRdData, size_t ArdLen) {
    assert(pRdData);
    memset(pRdData, 0, ArdLen);
    return pLo->spiWriteRead(RW_CONFIG, oBuf.data(), oBuf.length(),
                             pRdData, ArdLen);
}
```


いろいろ書きましたが……

不具合原因は

接触不良

でした。



押さえると動く！！



下駄を作成！！

安定動作 &
ラズパイもケースに入れられた！

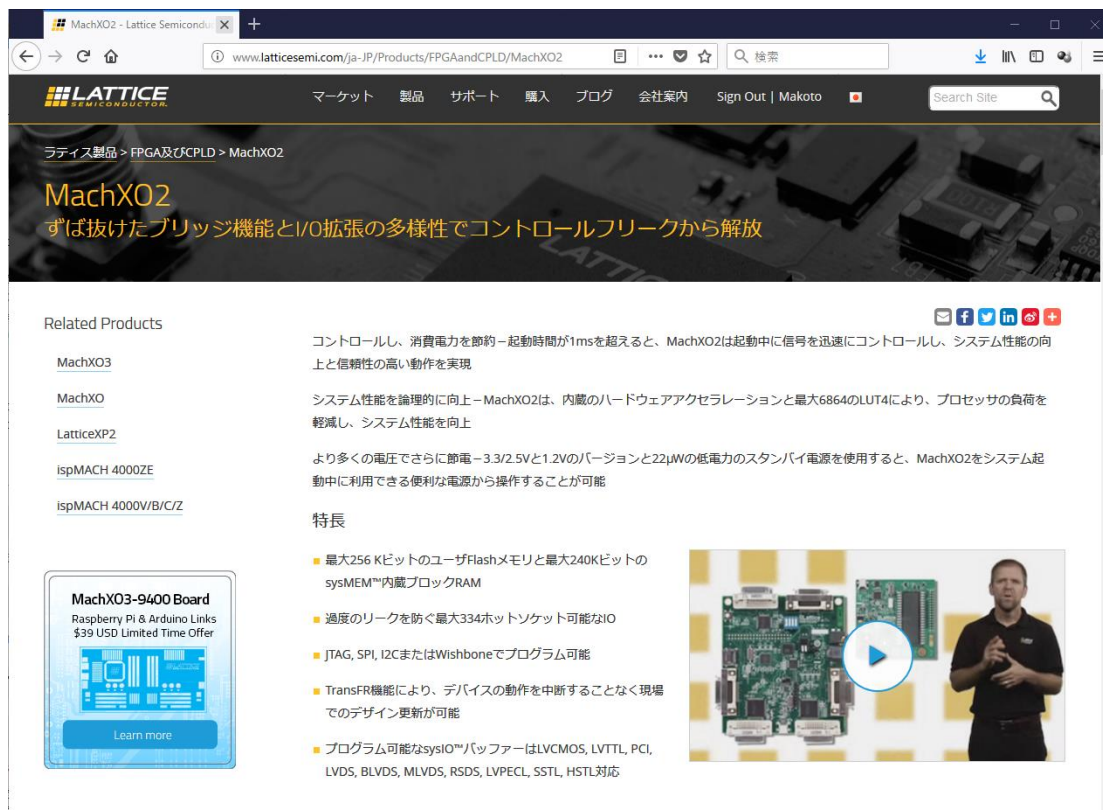
LatticeのFPGA開発環境

付属のプログラム経由で回路データを転送すれば動作しますが、
自分で回路を作成するためにはPCに開発用プログラムを入れる必要があります。

ボードに使用しているFPGAは

Lattice Semiconductor MachXO2-7000

Latticeのホームページで製品情報を見ると



MachXO2 - Lattice Semiconductor

www.latticesemi.com/ja-JP/Products/FPGAandCPLD/MachXO2

マーケット 製品 サポート 購入 ブログ 会社案内 Sign Out | Makoto

ラティス製品 > FPGA及びCPLD > MachXO2

MachXO2

ずば抜けたブリッジ機能とI/O拡張の多様性でコントロールフリークから解放

Related Products

- MachXO3
- MachXO
- LatticeXP2
- ispMACH 4000ZE
- ispMACH 4000V/B/C/Z

コントロールし、消費電力を節約 - 起動時間が1msを超えると、MachXO2は起動中に信号を迅速にコントロールし、システム性能の向上と信頼性の高い動作を実現

システム性能を論理的に向上 - MachXO2は、内蔵のハードウェアアクセラレーションと最大6864のLUT4により、プロセッサの負荷を軽減し、システム性能を向上

より多くの電圧でさらに節電 - 3.3/2.5Vと1.2Vのバージョンと22μWの低電力のスタンバイ電源を使用すると、MachXO2をシステム起動中に利用できる便利な電源から操作することが可能

特長

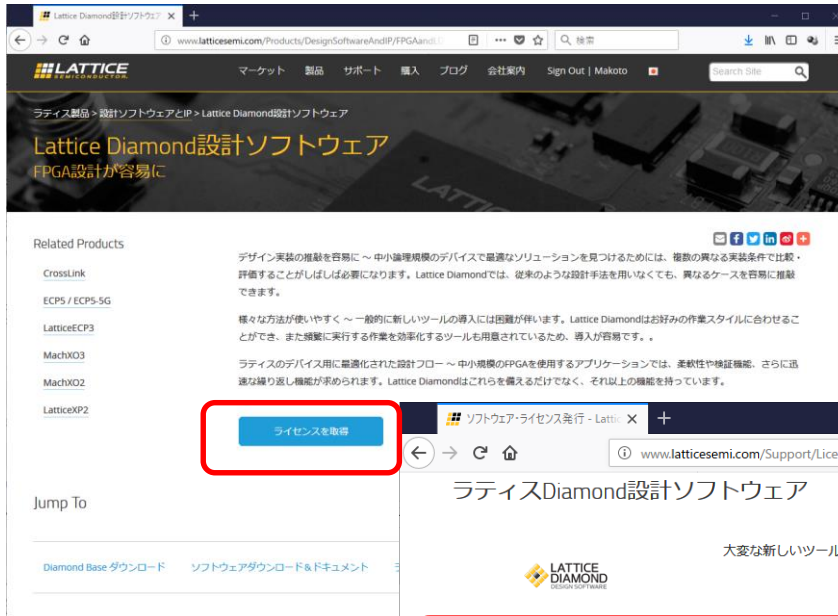
- 最大256 KビットのユーザーFlashメモリと最大240KビットのsysMEM™内蔵ブロックRAM
- 過度のリークを防ぐ最大334ホットソケット可能なIO
- JTAG, SPI, I2CまたはWishboneでプログラム可能
- TransFR機能により、デバイスの動作を中断することなく現場でのデザイン更新が可能
- プログラム可能なsysIO™バッファはLVCMOS, LVTTL, PCL, LVDS, BLVDS, MLVDS, RSDS, LVPECL, SSTL, HSTL対応

MachXO3-9400 Board
Raspberry Pi & Arduino Links
\$39 USD Limited Time Offer

Learn more

開発ツールは
Lattice Diamond

1. Diamondをダウンロード
2. インストール
3. 無料ライセンスを取得
(動作させるPCのMACアドレス必要)



ラティスDiamond無料ライセンス

ラティスのDiamond設計ソフトウェアは、コスト重視で、低消費電力のラティスFPGAアーキテクチャ向けに最適化された、業界をリードする設計と実装ツールを提供します。この無料ライセンスはユーザーに対応デバイスの非SERDESベースDiamondの設計と性能の評価を可能にします。

ライセンスの取得には次のものがが必要です

- 物理MACアドレス (12桁の16進値)

[無料ライセンスを取得する](#)

ラティスDiamond購読ライセンス

設計の探索が容易に。このフル購読ライセンスは、回避策に頼ることなく、ユーザーにすべての対応デバイスのDiamondソリューションの設計と最適化を可能にします。マルチコアマシン上で並行して実装することができ、最高のソリューションを迅速に見つけることができます。

ライセンスの取得には次のものがが必要です

- 物理MACアドレス (12桁の16進値) [詳細はFAQをご参照ください](#)
- Aldec USBキーシリアルナンバー (フローティングライセンスのみ)
- ラティスDiamondシリアルナンバー
- 以前にサポートポータルに接続したことのない方は、連絡先情報を確認する必要があります

Diamond LatticeECP3 Versa キット無料ライセンス

ラティスDiamond Versaキット無料ライセンスはユーザーにECP3 Versa開発ボードを使った設計の開発、検証、評価を可能にします。この開発キットは業界最安値のエッジシステムを構築するプラットフォームです。ECP5 Versaボード向けDiamondライセンスは、ボード付属の説明書にしたがって取得してください。

ライセンスの取得には次のものがが必要です

- 物理MACアドレス (12桁の16進値)

[物理MACアドレス \(12桁の16進値\)](#)

ラティスDiamond購読ライセンス5日間延長

デバイス設計と開発よりも速いスピードでやってくる購入サイクルによって機会を見逃さないでください。ライセンスの更新手続きの際、一時的なライセンスの延長が必要な場合でも大丈夫です。

このライセンスは、購読ライセンスが満了になった場合の一時的な延長のみ有効です。

ライセンスの取得には次のものがが必要です

- ラティスDiamondシリアルナンバー
- 以前にサポートポータルに接続したことのない方は、連絡先情報を確認する必要があります

[延長のリンク](#)

付属のサンプル回路を生成しました。

Lattice Diamond - Reports

File Edit View Project Design Process Tools Window Help

File List

- pif
 - LCMX02-7000HC-4TG144C
 - Strategies
 - Area
 - I/O Assistant
 - Quick
 - Timing
 - Strategy1
 - flasher
 - Input Files
 - ../common/flasher.vhd
 - ../common/piffla.vhd
 - Synthesis Constraint Files
 - ../common/pif2.lpf
 - LPF Constraint Files
 - ../common/pif2.lpf
 - Debug Files
 - Script Files
 - Analysis Files
 - Programming Files

Design Summary

 - Project
 - Project Summary
 - Process Reports
 - Synplify Pro
 - Map
 - Place & Route
 - Signal/Pad
 - Bitstream/JEDEC
 - Analysis Reports
 - Map Trace
 - Place & Route Trace
 - I/O Timing Analysis
 - Tool Reports
 - I/O SSO Analysis
 - Hierarchy Parsing Report
 - PIO DRC
 - TCL Command Log
 - ECO Editor Change Log

pif project summary

Module Name:	pif	Synthesis:	SynplifyPro
Implementation Name:	flasher	Strategy Name:	Strategy1
Last Process:		State:	
Target Device:	LCMX02-7000HC-4TG144C	Device Family:	MachXO2
Device Type:	LCMX02-7000HC	Package Type:	TQFP144
Performance grade:	4	Operating conditions:	COM
Logic preference file:	pif2.lpf		
Physical Preference file:	syn/pif_flasher.prf		
Product Version:	3.10.2.115	Patch Version:	
Updated:	2018/05/16 08:58:23		
Implementation Location:	C:/lsc/diamond/3.10_x64/bin/nt64/pif2/firmware/flasher/syn		
Project File:	C:/lsc/diamond/3.10_x64/bin/nt64/pif2/firmware/flasher/flasher.ldf		

Output

```
INFO - C:/lsc/diamond/3.10_x64/ispfpga/userware/NT/SYNTHESIS_HEADERS/machxo2.v(498,1-501,10) (VERI-9000) elaborating module 'IB_uniq_1'
INFO - C:/lsc/diamond/3.10_x64/bin/nt64/pif2/firmware/common/flasher.vhd(41,3-41,48) (VHDL-1400) back to vhdl to continue elaboration
INFO - C:/lsc/diamond/3.10_x64/bin/nt64/pif2/firmware/common/flasher.vhd(42,3-42,41) (VHDL-1399) going to verilog side to elaborate module GSR
INFO - C:/lsc/diamond/3.10_x64/ispfpga/userware/NT/SYNTHESIS_HEADERS/machxo2.v(494,8-494,11) (VERI-1018) compiling module GSR_uniq_1
INFO - C:/lsc/diamond/3.10_x64/ispfpga/userware/NT/SYNTHESIS_HEADERS/machxo2.v(494,1-496,10) (VERI-9000) elaborating module 'GSR_uniq_1'
INFO - C:/lsc/diamond/3.10_x64/bin/nt64/pif2/firmware/common/flasher.vhd(42,3-42,41) (VHDL-1400) back to vhdl to continue elaboration
INFO - C:/lsc/diamond/3.10_x64/bin/nt64/pif2/firmware/common/piffla.vhd(51,8-51,19) (VHDL-1067) elaborating pif_flasher_uniq_0(rtl)
INFO - C:/lsc/diamond/3.10_x64/bin/nt64/pif2/firmware/common/piffla.vhd(125,3-131,37) (VHDL-1399) going to verilog side to elaborate module osch
INFO - C:/lsc/diamond/3.10_x64/ispfpga/userware/NT/SYNTHESIS_HEADERS/machxo2.v(1793,8-1793,12) (VERI-1018) compiling module OSCH_uniq_1
INFO - C:/lsc/diamond/3.10_x64/ispfpga/userware/NT/SYNTHESIS_HEADERS/machxo2.v(1793,1-1798,10) (VERI-9000) elaborating module 'OSCH_uniq_1'
INFO - C:/lsc/diamond/3.10_x64/bin/nt64/pif2/firmware/common/piffla.vhd(125,3-131,37) (VHDL-1400) back to vhdl to continue elaboration
INFO - C:/lsc/diamond/3.10_x64/bin/nt64/pif2/firmware/common/piffla.vhd(12,8-12,19) (VHDL-1067) elaborating DownCounter_uniq_0(rtl)
INFO - C:/lsc/diamond/3.10_x64/bin/nt64/pif2/firmware/common/flasher.vhd(51,3-51,54) (VHDL-1399) going to verilog side to elaborate module OB
INFO - C:/lsc/diamond/3.10_x64/ispfpga/userware/NT/SYNTHESIS_HEADERS/machxo2.v(857,8-857,10) (VERI-1018) compiling module OB_uniq_1
INFO - C:/lsc/diamond/3.10_x64/ispfpga/userware/NT/SYNTHESIS_HEADERS/machxo2.v(857,1-860,10) (VERI-9000) elaborating module 'OB_uniq_1'
INFO - C:/lsc/diamond/3.10_x64/bin/nt64/pif2/firmware/common/flasher.vhd(51,3-51,54) (VHDL-1400) back to vhdl to continue elaboration
INFO - C:/lsc/diamond/3.10_x64/bin/nt64/pif2/firmware/common/flasher.vhd(52,3-52,52) (VHDL-1399) going to verilog side to elaborate module OB
INFO - C:/lsc/diamond/3.10_x64/ispfpga/userware/NT/SYNTHESIS_HEADERS/machxo2.v(857,8-857,10) (VERI-1018) compiling module OB_uniq_2
INFO - C:/lsc/diamond/3.10_x64/ispfpga/userware/NT/SYNTHESIS_HEADERS/machxo2.v(857,1-860,10) (VERI-9000) elaborating module 'OB_uniq_2'
INFO - C:/lsc/diamond/3.10_x64/bin/nt64/pif2/firmware/common/flasher.vhd(52,3-52,52) (VHDL-1400) back to vhdl to continue elaboration
INFO - C:/lsc/diamond/3.10_x64/ispfpga/userware/NT/SYNTHESIS_HEADERS/machxo2.v(498,1-501,10) (VERI-9000) elaborating module 'IB_uniq_1'
INFO - C:/lsc/diamond/3.10_x64/ispfpga/userware/NT/SYNTHESIS_HEADERS/machxo2.v(494,1-496,10) (VERI-9000) elaborating module 'GSR_uniq_1'
INFO - C:/lsc/diamond/3.10_x64/ispfpga/userware/NT/SYNTHESIS_HEADERS/machxo2.v(857,1-860,10) (VERI-9000) elaborating module 'OB_uniq_1'
INFO - C:/lsc/diamond/3.10_x64/ispfpga/userware/NT/SYNTHESIS_HEADERS/machxo2.v(857,1-860,10) (VERI-9000) elaborating module 'OB_uniq_2'
INFO - C:/lsc/diamond/3.10_x64/ispfpga/userware/NT/SYNTHESIS_HEADERS/machxo2.v(1793,1-1798,10) (VERI-9000) elaborating module 'OSCH_uniq_1'
Done: design load finished with (0) errors, and (0) warnings
```

Tcl Console Output Error Warning Info*

Ready

Mem Usage: 144,360 K

```

-----
-- flasher.vhd
--
-- Initial entry: 21-Apr-11 te
--
-- VHDL hierarchy is
--     flasher      top level
--     piffla.vhd   does the work!
--
-----
--
-- Copyright (c) 2001 to 2013  te
--
-----

library IEEE;      use IEEE.std_logic_1164.all;
library machxo2;    use machxo2.components.all;

=====
entity flasher is
    port ( GSRn      : in  std_logic;
          LEDR,
          LEDG       : out std_logic );
end flasher;

=====

architecture rtl of flasher is

    component pif_flasher is port ( red, green : out std_logic );
    end component pif_flasher;

    signal red_flash,
           green_flash : std_logic;

    signal GSRnX      : std_logic;
    attribute pullmode : string;
    attribute pullmode of GSRnX: signal is "UP"; -- else floats

begin
    -- global reset
    IBgsr  : IB  port map ( I=>GSRn, O=>GSRnX );
    GSR_GSR : GSR port map ( GSR=>GSRnX );

    -- LED flasher
    F: pif_flasher port map ( red      => red_flash,
                             green    => green_flash );

    -- drive the LEDs
    REDL: OB port map ( I=>red_flash , O => LEDR );
    GRNL: OB port map ( I=>green_flash, O => LEDG );

end rtl;
-- EOF -----

```